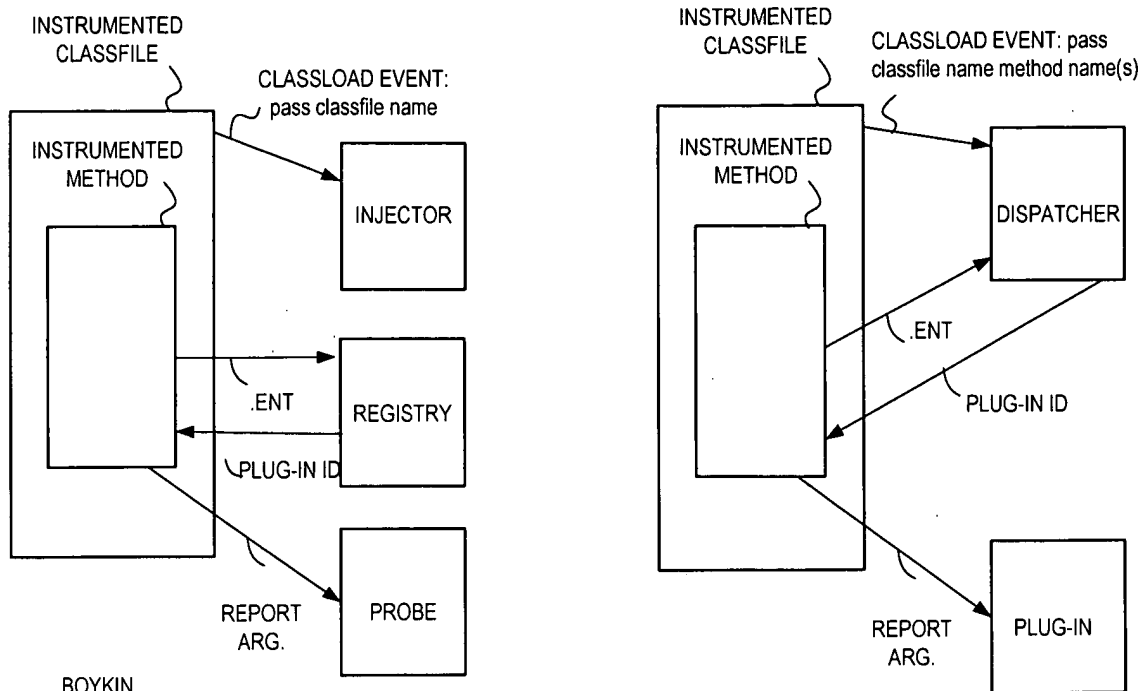


## **COMMENTS**

The enclosed is responsive to the Examiner's Office Action mailed on 11/16/07. At the time the Examiner mailed the Office Action claims 1-46 were pending. By way of the present response the Applicant has: 1) amended claims 1, 19, 36 and 37; 2) not canceled any claims; and. 3) not added any claims. As such, claims 1-46 remain pending. The Applicant respectfully requests reconsideration of the present application and the allowance of all claims 1-46.

The Examiner has maintained that independent claims 1, 19 and 37 are anticipated by Boykin. The figure below depicts the architecture of the Boykin reference compared against an architecture disclosed in the Applicant's specification.



Notably, the Boykin reference includes a Registry that is loaded pre-runtime with information identifying the methods within a classfile that need to be instrumented. This information is not received from a classfile - it provided by a user (See, Boykin para. 0048, 0049). During run-time when a particular class file is loaded ("Classload Event" in the above figure), the only information that is received from the classfile is the classfile's name (which is received by the Injector). The information concerning which methods within the classfile that need to be instrumented is already contained in the Registry, so the Injector, uses the class file name as a look up parameter to identify which methods are to be instrumented.

By contrast, in an architecture disclosed by the Applicant's specification, the

information that is received from the classfile during the classload event includes the classfile name and the names of instrumented methods. Thus, the following claim element that appears in each of independent claims 1, 19 and 37 is not disclosed by Boykin:

receiving from a classfile registration information comprising a class name and different method names for more than one of said class's methods

The Applicant has added an additional limitation indicating that a dispatcher receives the information from a classfile - however - this limitation is not necessary because Boykin simply does not disclose receiving a classfile name and method names from a classfile. This perspective is essentially the same offered by the Applicant in the Applicant's last response which is again repeated below in part:

As the Applicant understands the Boykin reference, the Boykin reference discloses a system in which a software developer, pre-runtime, defines "locations" within the program code where additional bytecode instructions are to be inserted during runtime when the classfile is loaded. These locations, as the Applicant understands them, may be defined by a method name (or constructor name) and the classfile to which the method (constructor) belongs. These locations are then recorded in a registry. During runtime, in order to instrument a classfile with additional bytecode instructions, the classfile passes some identifier of only itself to the registry. In response, the registry checks to see if any locations are to be

instrumented for the classfile that has just identified itself, and, if so, triggers the instrumentation of the classfile at the appropriate locations.

The following citations from Boykin demonstrate the accuracy of the above paragraph (additional emphasis being added).

"[T]he present invention comprises the technique of injecting probe hooks into code at runtime." Boykin, para. [0034]

"Hooks are directly injected into the class files at class-load time." Boykin, para. [0033]

"The user . . . specifies within the probe registry the Java elements that need to be instrumented . . . , i.e., the probe locations." Boykin, para. [0048].

"[E]ach probe is associated with a location in an application, e.g., a specific method within a specific class. The probes along with the associated locations are registered in a registry. . . . At class load time, an injector determines whether a loaded class has any instrumentation locations as predetermined by information in the registry." Boykin, Abstract.

"During the class load process, the class loader provides an indication, e.g., class load event notification 408, to injector 410, which then injects hooks into the classes." Boykin, para. [0045].

"Injector 410 can determine whether to inject a hook into a recently loaded class by querying the probe registry 416, e.g., by using an identifier of the recently loaded class, which may be provided to injector 410 through class load event notification 408 . . . . If the registry has at least one location for the recently loaded class, the injector proceeds to injector or embed at least one hook at an indicated method or constructor. If the registry lacks a location for the recently loaded class, then the injector does not modify the original class file . . . ." Boykin, para. [0046].

When the injector is notified that a new Java class is being loaded (step 702), it queries the registry to determine whether the newly loaded class needs to be instrumented, the injector then queries which methods, constructors, and fields within that class need to be instrumented (step 706). The injector then injects the hooks at the specified locations (step 708), thereby completing the process in the probe injection phase . . . ." Boykin, para. [0050].

Viewing the Boykin reference in a light most favorable to the Examiner's position, and without admitting to as much, the registry (206/416) of Boykin

corresponds to the Applicant's claimed "plug-in pattern" and the injector (216/410) of Boykin corresponds to a "dispatcher" described in the Applicant's specification that receives the notification event of a classfile. See, e.g., Applicant's specification, Figs. 17, 18, paras. [0143] - [0147]; [0153]. Briefly describing the subject matter concerning Figs. 17 and 18 of the Applicant's specification, the Applicant's specification describes a process where a classfile that has already been modified passes both classfile identity and method name information to the dispatcher.

Thus, the Applicant's claims are distinctive from Boykin for at least two reasons. First, whereas Boykin discloses that in order to trigger a look-up into the registry the identity of only a classfile is presented to the injector, by contrast, the Applicant's claims indicate that classfile identity and method names are presented [e.g., to a dispatcher]. Second, whereas Boykin discloses that the classfile is not yet modified when the classfile identity is passed to the injector, by contrast, the Applicant's claims indicate that the classfile is already modified when the classfile identity (and method names) are presented [e.g., to a dispatcher]. Therefore, the Applicant's claims are not anticipated by the Boykin reference.

In the further interests of efficiency, the Applicant reserves the right under MPEP 2144.03.C to cause the Examiner to find in the prior art subject matter to which the Examiner has taken Official Notice at a later time in the prosecution of the present case when the subject matter of such prior art is actually at issue.

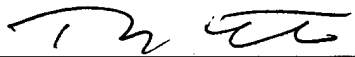
**REMARKS**

If there are any additional charges, please charge Deposit Account No. 02-2666. If a telephone interview would in any way expedite the prosecution of this application, the Examiner is invited to contact Robert B. O'Rourke at (408) 720-8300.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 2/19/08

  
\_\_\_\_\_  
Thomas C. Webster  
Reg. No. 46,154

1279 Oakmead Parkway  
Sunnyvale, CA 94085-4040  
(408) 720-8300